为什么要用git

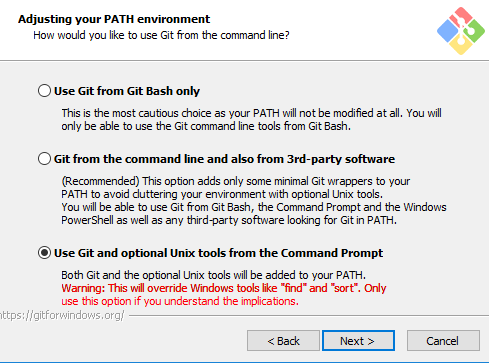
简单来说git就是一个版本管理工具，它可以精确记录到文档每一行修改，并且可以随时回退，于个人，可以管理本地工程化项目，同时还可以借助远程仓库，同步多端设备。即使是隐私数据文件，也可以在本地电脑搭建环境，无泄漏风险（远程仓库也支持私有仓库啦）。于团队，无需担心版本覆盖，无需担心同步修改，每个版本对应的修改记录以及相关负责人，甩锅无忧。

相对于其它版本管理工具，git最大优势在于它的分布式加持，什么文件损坏，磁盘损坏，黑客入侵，服务器宕机，IP封锁，贸易战爆发？放心，只要你托管到git，分布式的优势可以让你很顺便地做到两地三中心的备份，从此数据无忧。

百度一下git，可以很轻易找到git官网，虽然是英文的，但也可以很轻易找到两行Downloads 大字，下载无忧。



下载完成后，打开安装包，选好安装路径，你会发现还有一堆选项要你选，关键是全英文提示，放心，每个步骤都贴心给到你默认选项，傻瓜式next，安装无忧。



安装完成后，你发现桌面并没有任何变化，找不到打开方式，然而只要点击右键菜单，就会惊喜发现多了个git bash here 和 git GUI here选项，这就说明安装搭建完成，从此打开无忧。

选择git bash here 打开的是一个黑框框，这是一种基于命令的方式操作git，而选择git GUI here 打开的是图形化界面，图形化操作，但本篇文章花那么大篇幅讲一个图形化软件的使用，这不就很没意思了嘛，还是要有点逼格追求的，因此以下内容介绍的都是基于命令行的操作方式。

根据不同需求，这里列举了三种不同玩法以及进阶方式，话不多说，开始进行正式操作吧！

本地基础版

基础操作： 初始化 -> 本地暂存 -> 本地提交（定版）

本地初始化操作很简单，进入需要被git管理的目录，打开命令行窗口，输入git init ，回车后就完成啦！

如果你所在的文件系统默认可以看到隐藏文件的话，可以看到有个.git 文件夹，这就说明初始化成功啦！

初始化完成后，你就可以畅所欲言地修改了，当然，如果你那目录本身有文件，建议先提交一个初始化版本再下手，避免改错回不去了。那接下来看怎么来提交。

再次使用命令行工具，使用git add . 暂存本地所有文件。

暂存完成后，再次使用命令行工具，输入git commit –m “[双引号内部填上提交版本记录信息]”，即可完成本地提交操作，怎么样，简单吧？

如果这么简单的操作觉得没问题的话，就不妨来了解一下它背后的原理和理论吧：

Git 作为本地版本库，为文件的修改提供了三重记录防护，分别体现在三个区域：工作区/暂存区/本地仓库。所谓工作区其实就是本地文件夹，平时对某个文件任何修改都是在工作区做记录，对于一般操作者来说，没有git管理的工作区，如果工作区改错了某些内容，大多是使用Ctrl + Z ZZZZZ 疯狂撤回到某个正确的版本。这种方式其实很不灵活，首先不是所有软件都支持无限次的Ctrl + Z的，即使支持，如果文件被重新打开了，或者只想撤回某一部分的记录，都很不方便操作。使用git你就无需担心这个问题。

在说暂存区之前，先来说本地仓库，本地仓库一般作为确定定版的版本，当然这不是唯一的标准，需要根据项目规模来定，项目规模大的话，把整个项目完成了再提交显然不合适，一般是拆分成多个小模块，提交每一个小模块完成的部分。总之也就是说，提交上的版本，一般是很少再回来修改的。

因此暂存区，就是工作区到本地仓库之间的过渡区，总有些时候，你完成了某个模块中某个文件的修改，但你不确定接下来的工作还要不要再补充点什么的时候，一直在工作区看着有点烦，这时就可以放进暂存区。

上面介绍的这么多操作，其实就是不同区之间的进阶，git add 操作能够使工作区的文件放进暂存区，git commit 操作能够使暂存区的文件提交到本地仓库。这个流程一个都少不了，并不存在直接将工作区的文件提交到本地仓库的操作！

本地进阶版：

前面说到git最大的魅力在于它的版本穿梭能力，基础版并没给出具体演示，那就这楼来看看吧：

要想版本穿梭，势必得知道你做过哪些操作吧？也势必得知道有哪些版本可以穿梭吧？所以“查询”命令是在此之前不可缺少的部分：

查看当前工作目录的文件修改状态：git status

查看某个文件修改记录： git diff

查看版本提交记录：git log 版本太多太长的时候，我们并不需要看那么多信息，希望以简要的方式查看：git log --pretty=oneline

撤回工作区的文件修改记录：git checkout --<file> 其中file 指向带路径文件。

将暂存区文件撤回到工作区：git reset HEAD <file> 可以将暂存区中的文件撤销掉，又名 unstage

撤销已提交的版本：git reset --hard <commitId>：又名回退（reset）需要注意这是硬回退模式，这种回退方式存在一定风险，会将你在本地以及暂存内的所有修改撤销，使用该命令前请确保本地工作区是干净的，否则推荐使用软模式（- - soft）或混合模式（- - mixed）（默认）

联机同步版：

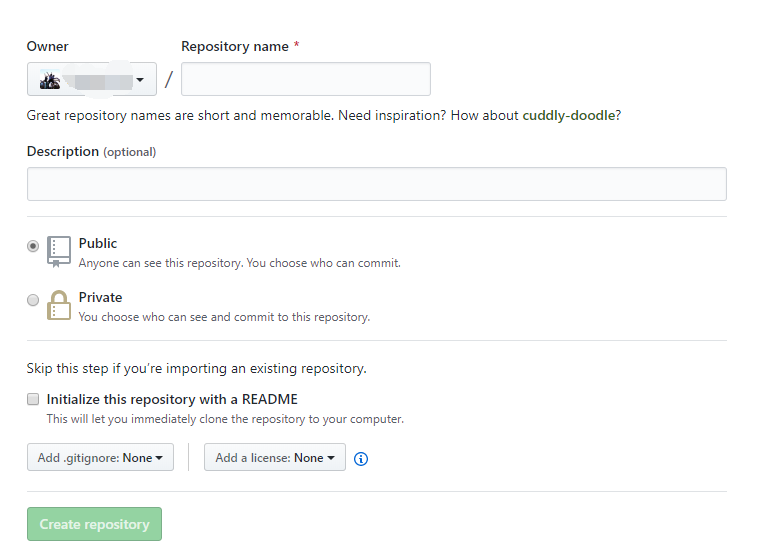
前面所做的操作不过都是本地电脑内部瞎嗨，并不能实现多机同步，不过要想实现同步，就必须要借助git 的远程仓库，也就是要接入互联网啦。常用的远程仓库包括知名的github、国内的gitee以及用于企业内部的gitlab。三种仓库功能和操作都差不多，根据自身的需要选择一到两个仓库来用即可。百度进入它们的官网，通过邮箱注册一个账号就完成啦！

接入远程仓库，麻烦就麻烦在第一次配置，配置完成后后续操作都非常方便了。

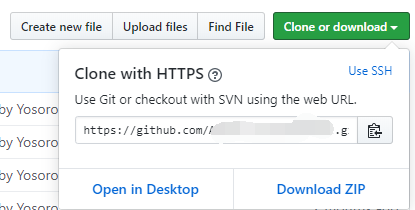
首先要配置的是本地用户名和邮箱：git config user.name “your name” git config user.email “[youremail@email.com](mailto:youremail@email.com)” 这个操作本应该在commit 操作之前，或者说在git 安装完成后就得配置了。在本地配置的用户名和邮箱最好要保持和远程仓库注册的一致，否则同步上去就变成未知用户了。

接下来还要在官网上创建一个Repositories ，以github 为例，它的创建界面是这样的：

填入仓库名，描述，选择开放还是私有仓库即可创建啦。



远程仓库创建完成后，就能拿到仓库的地址啦，



远程仓库的地址有Https 和 ssh 两种方式，这里为了快速上手，先以https为例，但是为了操作方便、更高的安全性、以及更好的授权，强烈建议有点追求的小伙伴，使用ssh的方式，虽然多了点配置，但这都会是一劳永逸的效果。

拷贝远程仓库的https地址，在本地git bash中输入命令：git remote add origin [https://github.com/xxxx.git] （其中最后一串是远程仓库的地址）就算是将本地仓库和远程仓库建立关联啦！

搞定以上配置，接下来就是常规操作啦，这里假设一个场景，当你想将自己本地仓库研究的成果拿到团队去分享，首先要做的就是将本地仓库同步到远程仓库，又名推送（push），使用命令：git push origin master ，输入注册远程仓库的账号密码，即可完成啦。（输入账号密码的操作很麻烦而且有安全风险，因此还是建议使用ssh方式）

关于origin 和 master 其实有还涵盖了另一门学问，但对于刚新手来说，可以先忽略这里面高深的内容继续往下操作。

接下来在团队的新机器里，并没有任何你本地仓库的信息，这时就需要将远程仓库同步到这台新机器，这种同步操作又名克隆（clone）： git clone [https://github.com/xxxx.git] （其中最后一串是远程仓库的地址）。这样就完成了在空的目录上新建了和远程一样的git仓库了。

接下来呢，你的队友提出了一堆修改意见，你当即在那台新机器修改完成了，接下来需要的操作当然也是同步到远程，也就是进行提交并推送的操作。

但是当你回到自己本地的机器后，又如何将新机器上所做的修改更新到本地呢？这时候的更新就不能再是克隆了，而应该是拉取（fetch），使用的命令：git pull origin master。